An Introduction to Information Retrieval using Singular Value Decomposition and Principal Component Analysis

Tasha N. Underhill

April 12, 2007

It takes only a quick Google search to realize the breadth of applications of linear algebra. Heat distribution, Markov chains, Sociology, Games, Cryptography...the list goes on and on. But wait, even the search engine itself stems from an application of linear algebra! Books could be written about all of these topics, but in this paper we will focus on two methods of information retrieval which rely heavily on linear algebra. The first is the topic of information retrieval (IR), for which we will outline the basic vector space model and explain how rank-reduction of the term-by-document matrix minimizes cost and improves efficiency of the retrieval. This model is quite useful for even very large dimensional databases of objects indexed by key words. A second method of information matching and retrieval may be applied to large databases of similar graphics. A statistical method, Principal Component Analysis (PCA) is often used to classify and recall images with particular application in face recognition and medical imaging.

SVD and the Vector Space Model

In this age of technology, there is a need for fast and efficient data retrieval. This is applicable in small businesses, libraries and on a grand scale, the internet. One way this issue is approached uses a matrix to represent large amounts of data, then relies on vector operations to extract a set of data which is close to the users query. We outline the simplest method, the vector space model using an example:

In a (very small) database of cook books there are 5 documents, titled:

d1: Quick and Easy Dinners
d2: Vegetarian Cooking: Healthy Meals Made Easy
d3: Cooking Up a Storm: Preparing Meals for 100+
d4: 101 Healthy Crockpot Dinners
d5: Cook Like Martha: 50 Recipes Guaranteed to Please

Each of these documents may be indexed by certain terms contained in the titles. The associated terms for this set of documents are:

t1: meal
t2: dinner(s)
t3: recipe
t4: cook(ing)
t5: health(y)
t6: vegetarian

To allow for variations in the user's query, *stemming* is used to improve the return, such that words are reduced to their stems. For instance, a user may search for **cooking**, which will be recognized as **cook**. The indexed documents may be displayed as a 6x5 (term x document) matrix A, where the element a_{ij} refers to the number of times a term i occurs in the title of a document j.

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

We now have a nice matrix which tells us there are certain terms in the title of certain documents. However, since the content of a document is not determined by the number of times a term appears but rather the relative frequencies of the terms, we scale each column so it has norm 1:

$$\hat{A} = \begin{bmatrix} 0 & 0.5 & 0.7071 & 0 & 0 \\ 1 & 0 & 0 & 0.7071 & 0 \\ 0 & 0 & 0 & 0 & 0.7071 \\ 0 & 0.5 & 0.7071 & 0 & 0.7071 \\ 0 & 0.5 & 0 & 0.7071 & 0 \\ 0 & 0.5 & 0 & 0 & 0 \end{bmatrix}$$

Let's see what happens when a user conducts a search. The user may wish to search for books relating to **healthy vegetarian dinners**. Representing this query as a vector with nonzero terms corresponding to the terms **healthy, vegetarian**, and **dinners** we have:

$$q = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

In order to relate the matrix A and the query vector q, we determine the cosine of the angle between q and each column a_j of A. The cosine is computed according to:

$$\cos\theta_j = \frac{a_j^T q}{\|a_j\|_2 \|q\|_2} = \frac{\sum_{i=1}^{t} a_{ij} q_i}{\sqrt{\sum_{i=1}^{t} a_{ij}^2} \sqrt{\sum_{i=1}^{t} q_i^2}} \tag{1}$$

The cosines between the angles are, in order, 0.57735, 0.288675, 0, 0.57735, 0. This means the documents which are returned are d1, d2 and d4. The documents which contain none of the query terms are accurately not returned. One question this method brings up concerns the accuracy of the return: At what value should we place the cutoff for the cosine such that documents which do not meet the cutoff are not returned? In this very small example, we could set the cutoff to be $\cos\theta_j = 0.5$. This would mean only the documents "Quick and Easy Dinners" and "101 Healthy Crockpot Dinners" are returned. The only document containing the search term **vegetarian** is not returned, which is probably the most relevant to the user. It is easy to see the limitations of this IR model.
There are several methods which make this model more accurate and efficient. One method

uses the singular value decomposition and rank-reduction of the term-by-document matrix $\hat{A}$. In the singular value decomposition, $\hat{A} = UDV^T$, where U is a unitary mxm matrix. D is mxn matrix with the k singular values of $\hat{A}$ on the first k diagonal entries of D and the rest of the entries equal to zero. V^T is another unitary nxn matrix. For steps on how to compute a singular value decomposition, see [6], or employ the use of mathematics software such as *Mathematica*.
In this particular case, we have:

$$\hat{A} = \begin{bmatrix} 0 & 0.5 & 0.7071 & 0 & 0 \\ 1 & 0 & 0 & 0.7071 & 0 \\ 0 & 0 & 0 & 0 & 0.7071 \\ 0 & 0.5 & 0.7071 & 0 & 0.7071 \\ 0 & 0.5 & 0 & 0.7071 & 0 \\ 0 & 0.5 & 0 & 0 & 0 \end{bmatrix} =$$

$$\begin{bmatrix} 0.479 & 0.189 & 0.398 & -0.474 & -0.134 & -0.577 \\ 0.283 & -0.86 & -0.26 & -0.311 & 0.132 & 0 \\ 0.213 & 0.138 & -0.664 & 0.395 & 0.074 & -0.577 \\ 0.692 & 0.327 & -0.266 & -0.079 & -0.060 & 0.577 \\ 0.353 & -0.313 & 0.414 & 0.684 & -0.371 & 0 \\ 0.203 & 0.035 & 0.303 & 0.218 & 0.904 & 0 \end{bmatrix} \begin{bmatrix} 1.457 & 0 & 0 & 0 & 0 \\ 0 & 1.297 & 0 & 0 & 0 \\ 0 & 0 & 0.837 & 0 & 0 \\ 0 & 0 & 0 & 0.632 & 0 \\ 0 & 0 & 0 & 0 & 0.306 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0.194 & 0.593 & 0.568 & 0.308 & 0.439 \\ -0.663 & 0.092 & 0.281 & -0.639 & 0.254 \\ -0.31 & 0.508 & 0.112 & 0.131 & -0.785 \\ -0.492 & 0.275 & -0.619 & 0.417 & 0354 \\ 0.431 & 0.554 & -0.45 & -0.552 & 0.032 \end{bmatrix}$$

To reduce the rank of $\hat{A}$ to k, choose the first k nonzero singular values of $\hat{A}$ and their corresponding columns of U and of V^T. We know the rank of $\hat{A}$ is $r_{\hat{A}}$=5 because A has five nonzero singular values [6]. Maybe we should reduce the rank to 4 or even 3. To determine the amount of change that would occur by reducing the rank of $\hat{A}$, the Frobenius norm of $\hat{A}$ minus the rank-reduced $\hat{A}_k$ compared to the Frobenius norm of $\hat{A}$ is used. Frobenius norm is defined as:

$$||A||_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2} \tag{2}$$

We are interested in finding the percentage of relative change of a rank-reduced matrix to determine by how much we may reduce without loss of information. This is done by the formula:

$$\frac{||\hat{A} - \hat{A}_k||_F}{||\hat{A}||_F} \tag{3}$$

The numbers associated with our example are:

$$\frac{||\hat{A}-\hat{A}_4||_F}{||\hat{A}||_F} = 0.13689,$$

for a rank-4 approximation to $\hat{A}$ and

$$\frac{||\hat{A}-\hat{A}_3||_F}{||\hat{A}||_F} = 0.702595,$$

for a rank-3 approximation. We see that reducing $\hat{A}$ to a rank 4 only requires about a 14% change, while reducing $\hat{A}$ further to rank 3 requires a massive 70%. Let's stick with a reduction to rank 4 and see what this buys us. As before, we must compare the query vector q to the rank-reduced A_4. Again we find the cosine of the angle between A_4 and q, and use a cutoff value which serves to display only relevant items. We use the same formula as before, but with transposed columns of A_4 rather than transposed columns of $\hat{A}$:

$$\cos\theta_j = \frac{(A_{k_j})^T q}{\|A_{k_j}\|_2 \|q\|_2} \tag{4}$$

Using this formula, the cosine of the angles between q and each column representing a document of A_4 are 0.53134, 0.519757, 0.0533917, 0.894272, and -0.003775. When we compare these values and the values obtained for the cosine of the angle between q and $\hat{A}$, (0.57735, 0.288675, 0, 0.57735, 0), we see a similarly high value for the cosine of the angle between the vectors for the first, second and fourth document. At first glance there doesn't seem to be much value in reducing the rank of the weighted term by document matrix. However, recall that the second document would not have been returned if a cutoff value was set at a reasonable $\cos\theta_j = 0.5$. Since the search terms included the term **vegetarian**, the second document is definitely relevant.
Besides a more precise return, rank-reduction also allows for cheaper and faster computing. We can rewrite equation (4) in the following way:

$$\begin{aligned}
\cos\theta_j &= \frac{(A_{k_j})^T q}{\|A_{k_j}\|_2 \|q\|_2} \\
&= \frac{(U_k D_k V_k^T e_j)^T q}{\|U_k D_k V_k^T e_j\|_2 \|q\|_2} \\
&= \frac{e_j^T V_k D_k (U_k^T q)}{\|D_k V_k^T e_j\|_2 \|q\|_2}
\end{aligned}$$

We use columns of the identity matrix, e_j to give the jth column of A_k. Now we define a vector $p_j = D_k V_k^T e_j$ and rewrite the formula for cosine as:

$$\cos\theta'_j = \frac{p_j^T (U_k^T q)}{\|p_j\|_2 \|q\|_2} \tag{5}$$

This representation of the cosine is especially good, because it makes it easy to relate the bases of U and V^T. The elements in the vector p_j are the coordinates of the corresponding column of A_k in the basis consisting of the columns of U_k. Further, based on observations about the projection of q into the column space of A_k, we can write an alternate cosine formula to compare query vector and document matrix [5]:

$$\cos\theta'_j = \frac{p_j^T(U_k^T q)}{\|p_j\|_2 \|U_k^T q\|_2} \tag{6}$$

Equation (6) needs only computations in a k-dimensional space, thus saving cost of computation.
We have shown a very crude model for data retrieval. Of course, there are many ways to improve and add to the complexity of this IR model. As this is merely an overview, the reader is referred to the list of references for more in-depth articles. Keeping with methods of information retrieval, we will now provide an overview of the use of Principal Component Analysis in reference to data storage and image retrieval.

PCA Applied to Image Recognition and Recall

Principal component analysis is a statistical method which may be used to find patterns in high-dimensional data in efforts to find a lower-dimensional representation of the data. Before we explore the application of PCA we must explain some basic statistics formulas, including standard deviation, variance and covariance. We assume a knowledge of eigenvalues and eigenvectors of a matrix, and then proceed to the method of PCA.
A set of data $X = \{x_1, x_2, ..., x_n\}$, with a mean $\bar{X}$, has a certain spread from the mean which is given by the standard deviation of the data set, s.

$$s = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{(n-1)}} \tag{7}$$

The variance of a data set is merely the square of the standard deviation, and is denoted s^2. As the variance and standard deviation only apply to one-dimensional data sets, they are not useful for comparing two ore more different sets of data. Thus, a measure is needed to determine how much the data in multiple dimensions vary from the mean with respect to each other. This is known as the covariance and is formulated as

$$cov(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)} \tag{8}$$

with Y denoting a set of data $Y = \{y_1, y_2, ..., y_n\}$. Covariance may only be measured between two dimensions. When dealing with higher dimensional data sets, a covariance matrix depicts the covariance between each dimension. The entries of a covariance matrix representing n dimensions are given by

$$c_{ij} = cov(x_i, x_j) \qquad 1 \leq i, j \leq n \tag{9}$$

The resulting covariance matrix is symmetric with the variance of each dimension on the diagonal. With the statistics we need in place, we may now discuss the method of PCA. To begin, the mean of a set of data must be subtracted from each of the elements in the set. Next calculate the covariance matrix. As this matrix is an nxn real symmetric matrix, we can compute its (positive) eigenvalues and eigenvectors. These eigenvectors are guaranteed to form an orthogonal set, thus in a two-dimensional example we would be able to plot the data and see that one eigenvector points along the line of the correlation of data, and the other would lie perpendicular to the first. [1][6] In higher dimensions, the eigenvector which corresponds to the largest eigenvalue is denoted the *principal component* of the data set. The principal component of a data set corresponds to the most significant relationship between the sets.

Now we are free to order the eigenvectors in descending order, according to their associated eigenvalues. The lower the eigenvalue, the less significant is the corresponding eigenvector. Next, we may reduce the dimension by omitting the least significant eigenvectors for the next step. The amount of reduction is determined heuristically, depending on the data set and conditions on its accuracy. The more we reduce the rank, the less the data will resemble the original. To give some perspective, in a very large dimensional space (such as when working with images), we may reduce the rank by several thousands in order to see a measurable difference from the original data. Once we have decided which eigenvectors we wish to keep, we form a *feature vector* which is a matrix with columns made up of the remaining k eigenvectors.

In order to create a new set of data in terms of the remaining eigenvectors, we use the feature vector and the original mean-adjusted data. This is done by matrix multiplication of the matrix V representing the transposed feature vector by the matrix R whose ith column lists the ith data point from each of the n dimensions of the set. The resulting matrix relates the ith data point ($1 \leq i \leq n$) to the set of $k \leq n$ eigenvectors of the covariance matrix.

If we wish to apply principal component analysis to information retrieval we find one natural application in image retrieval. One of the most interesting components of image retrieval is computerized facial recognition. In this application, a large collection of pictures of faces is used as a training set. Each face is indexed by pixel value and any new face is compared to a basis of the training set's *eigenfaces*. The following techniques are often employed in medical imaging where there may be a large volume of very similar images (such as brain scans).

Each of the M images is denoted by a $1xn$ vector which is the concatenation of pixel values of the n rows of the (gray-scale)image. As above, we must subtract the mean of the data from each face vector, which leaves us with the set of vectors $\{\Phi_1, \Phi_2, ..., \Phi_M\}$. Next we define the covariance matrix

$$\begin{aligned} C &= \frac{1}{M} \sum_{i=1}^{M} \Phi_i \Phi_i^T \\ &= AA^T \end{aligned}$$

where the column A_j=Φ_j, $1 \leq j \leq M$. We must find the eigenvalues and associated eigenvectors of AA^T, which are denoted as the set's eigenfaces and span a subspace M' of n^2. Now that we have the eigenface decomposition in place we may use it for facial recognition. When confronted with a new image Γ, the vector must be projected into "face space" by a weighting

$$\omega_k = \mathbf{u_k^T}(\Gamma - \Psi) \qquad\qquad 1 \leq k \leq M', \tag{10}$$

where $\mathbf{u_k^T}$ is an eigenvector of C, and Ψ is the mean of the data set. When placed as columns of a matrix, these weights ω_k form a matrix Ω^T, which describes how much contribution each eigenface makes to the new face. Ω is then compared using the distance between Ω and a pre-set class of faces described by Ω_k by calculating the Euclidean distance $\epsilon = \|(\Omega - \Omega_k)\|$ and determining if the value is below some threshold value α_ϵ. If $\epsilon \leq \alpha_\epsilon$ then the new face in question has similar enough characteristics to be classified as "known". Otherwise it is "unknown" and is not in the set of initial faces.
Some limitations of this application is the time it takes to calculate whether a new face is in a certain face class or not. Also, different lighting, movement or facial expressions will influence whether the face in question matches a known face in a database. Work has been done to apply PCA in order to aid the indexing of medical images such as brain scans, when there are many thousands of images which look very similar to the human eye. It is much more time-consuming for a human to analyze and index each brain scan by its features than for a computer to find which classification the scan is closest to, as determined by the Euclidean distance between two matrices.

While linear algebra plays a major role in simple information retrieval models, there are many more models which rely on other branches of mathematics. Vector space models often use matrices and matrix-vector operations to compartmentalize data, and exploit the geometric properties associated with the vectors to determine the proximity of query and relevant data in a set. The singular value decomposition is only a small example of a multitude of decompositions and factorizations related to vector space models. Effective models try to combat both *polysemy*, words which may have more than one meaning and *synonymy*, different words which mean the same thing. Both polysemy and synonymy must be considered when indexing documents as well as building a computerized information retrieval system. With very large databases, such as a collection of web pages on the internet or a library index, there is also a concern of cost-effectiveness of the IR system. Thus models are sought which reduce the number of calculations which must be performed for each search. The problems of indexing higher-dimensional objects such as images or video are combated with the use of dimension-reduction procedures. Principal component analysis allows very large sets of high-dimensional data to be accurately and efficiently described in much smaller dimension. The problems associated with indexing a large number of similar images, such as in medicine or astrophysics, may be a dealt with by an automated indexing which employs methods such as PCA. The science of information retrieval is a thriving subject, and will continue to improve the methods for indexing and retrieving documents from increasingly large databases.

Bibliography

[1] Smith, Lindsay I., "A tutorial on Principal Components Analysis,", (New Zealand: University of Otago, Feb. 26, 2002).

[2] Rechtsteiner, Andreas; Rocha, Luis M.; Wall, Michael E., "Singular Value Decomposition and Principal Component Analysis." A Practical Approach to Microarray Data Analysis. Ed. D.P. Berrar, W. Dubitzky, M. Granzow. (Norwell, MA: Kluwer, 2003): 91-109

[3] Turk, Matthew A.; Pentland, Alex P., "Face Recognition Using Eigenfaces, " *Vision and Modeling Group, The Media Laboratory,* (Boston, MA: Massachusetts Institute of Technology, 1991).

[4] Sinha, Usha; Kangarloo, Hooshang., "Principal Component Analysis for Content-based Image Retrieval," *RadioGraphics,*(22), (California: 17 Apr. 2002): 1271-1289.

[5] Berry, Michael W.; Drmac, Zlatko; Jessup, Elizabeth R., "Matrices, Vector Spaces, and Information Retrieval," *SIAM Review,***41**(2), (Jun. 1999): 335-362.

[6] Beezer, Robert A., "A First Course in Linear Algebra," (Tacoma, WA: Department of Mathematics and Computer Science, 2006).

[7] Berry, Michael W.; Browne, Murray, Understanding Search Engines: Mathematical Modeling and Text Retrieval. (Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999).